آکادمی ربوتک

دوره تنسورفلو پیشرفته

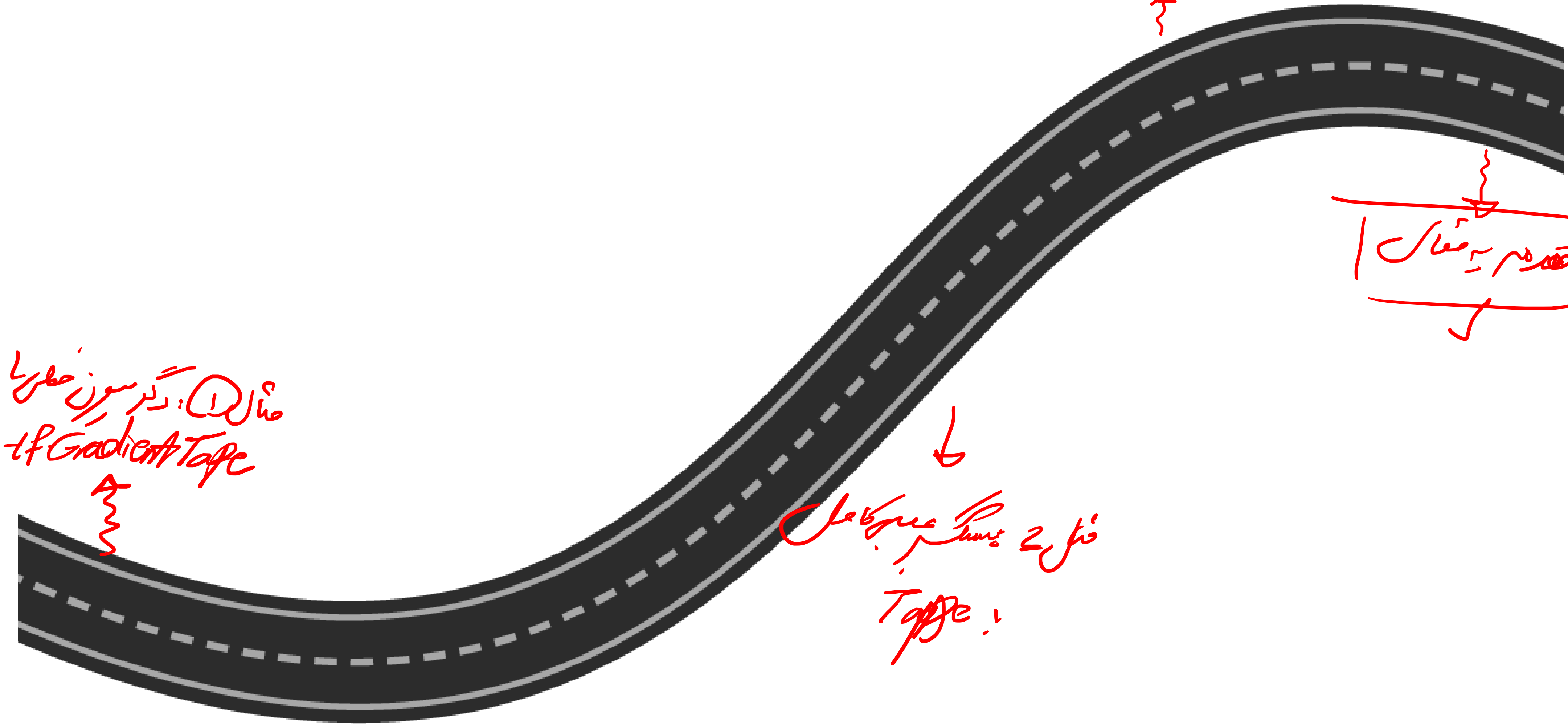
جلسه ششم : Custom Training Loop

دی شیخ با چراغ همی گشت گرد شهر
کز دام و دد ملولم و انسانم آرزوست

زاین همرهان سست عناصر دلم گرفت
شیر خدا و رستم دستانم آرزوست

گفتم که یافت می نشود جسته‌ایم ما
گفت آنچه یافت می نشود آنم آرزوست

چه گفته ایم و چه خواهیم گفت:



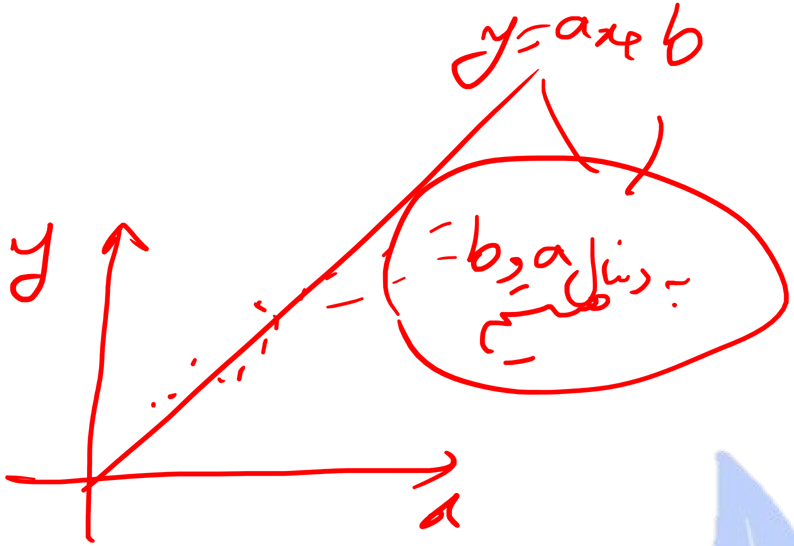
model subclassing

مقدمه و مثال

مثال 2 پیشرفته‌تر: شبکه عصبی کامل
با Tape

مثال 1: رگرسیون خطی با
tf.GradientTape

حل مساله رگرسیون خطی :

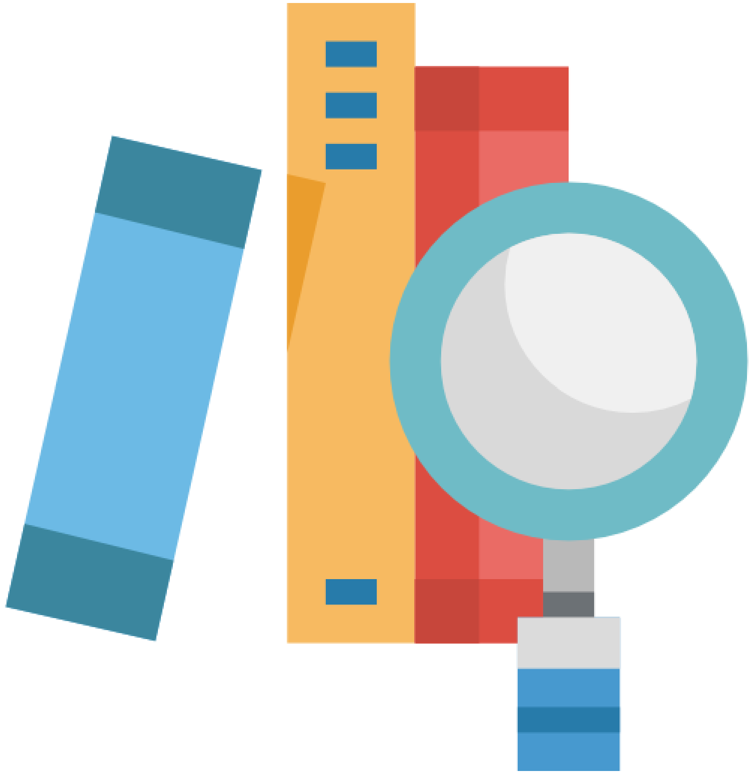


$y = ax + b$

به دنبال a و b هستیم

اضافه کردن کتابخانه های مورد نیاز

```python
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"
import tensorflow as tf
import matplotlib.pyplot as plt
```



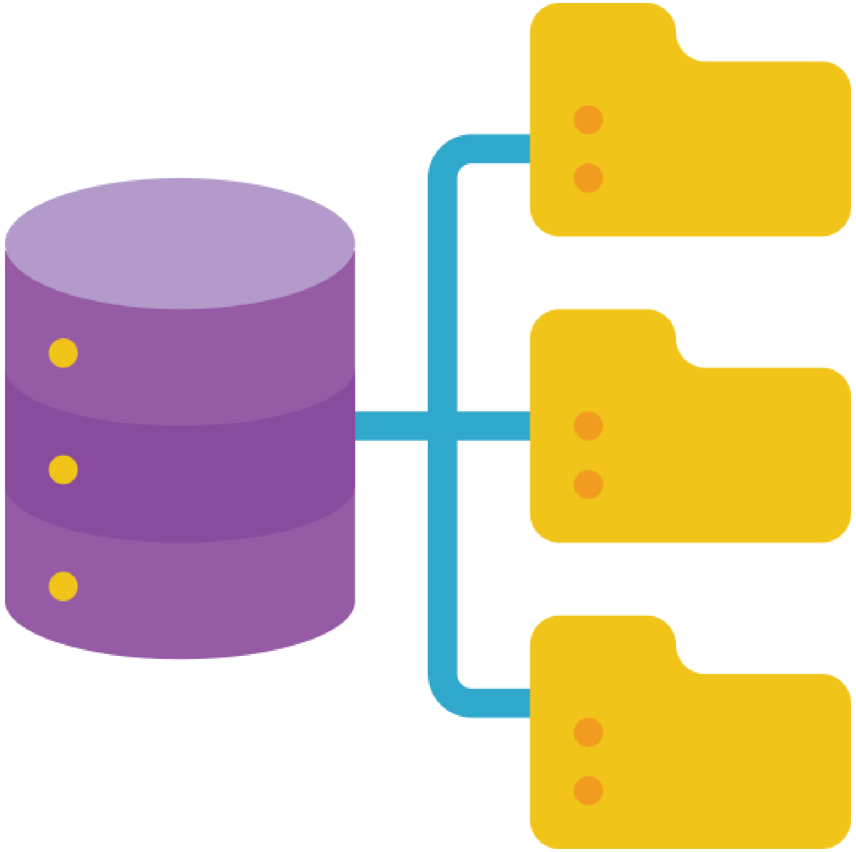
تعریف متغیرها و ثابت های برنامه

$\boxed{y = wx + b}$

مقادیر

```
w = tf.Variable(2.0)

b = tf.Variable(1.0)

TRUE_w = 3.0

TRUE_b = 2.0

NUM_EXAMPLES = 1000
```

ایجاد دیتاست



```
xs = tf.random.normal(shape=(NUM_EXAMPLES,))

ys = (TRUE_w * xs) + TRUE_b
```

واقع

رسم داده ها

```python
def plot_data(inputs, outputs, predicted_outputs):
  plt.scatter(inputs, outputs, c='b', marker='.', label = "Real Data")
  plt.scatter(inputs, predicted_outputs, c='r', marker='+', label = "Predicted Data")
  plt.legend()
  plt.show()

predicted_outputs = w*xs + b
plot_data(xs, ys, predicted_outputs)
```

مقادیر اولیه
$w = 2$
$b = 1$

y

مقادیر [illegible]
w
b

$|y|$

محاسبه Loss قبل از Train شدن

```
def loss(predicted_y, target_y):
  return tf.reduce_mean(tf.square(predicted_y - target_y))

print('Current loss: {:.2f}’ .format( loss(predicted_outputs, ys).numpy()))
```

تابع اصلی Train

سه تا کار (training) به عهده آن پیش

لیبل واقعی

α

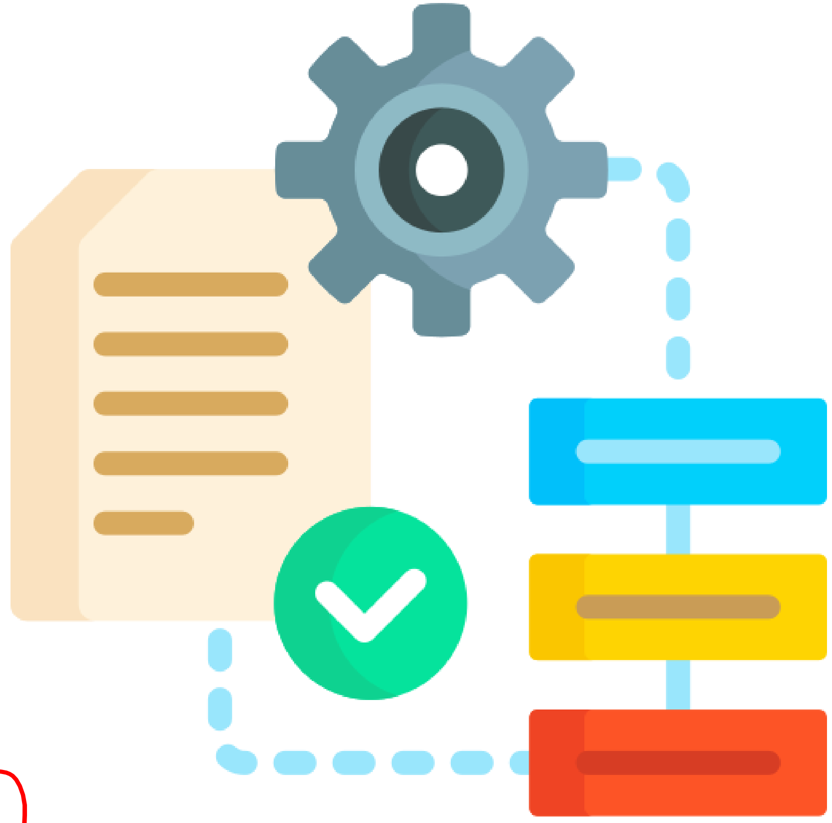
```
def train(outputs, learning_rate):
  with tf.GradientTape() as t:
    pred_y = w * xs + b
    current_loss = loss(pred_y, outputs)
  dw, db = t.gradient(current_loss, [w, b])
  w.assign_sub(learning_rate * dw)
  b.assign_sub(learning_rate * db)

  return current_loss
```

پیش بینی y

loss

$$\frac{\partial loss}{\partial w}$$

$$w^{+} = w^{-} - \alpha \frac{\partial L}{\partial w}$$



یاد آوری فرمول Gradient Descent

$$\underline{w_{t+1}} = \underline{w_t} - \alpha \boxed{\frac{\partial L}{\partial w_t}}$$

$\frac{\partial L}{\partial w} \rightarrow$

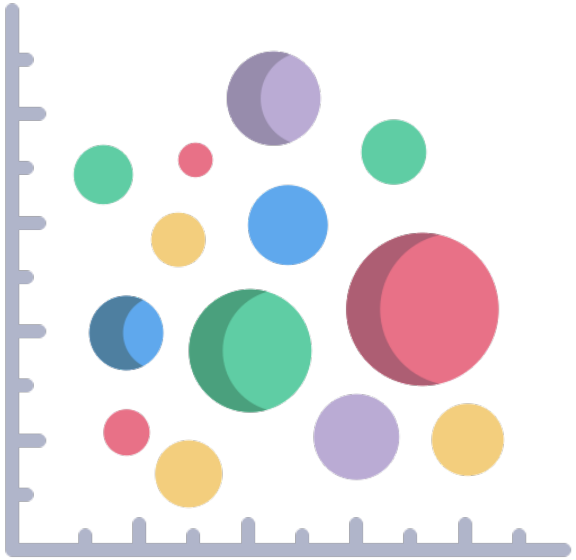
Train و ذخیره مقادیر وزن و بایاس

```
list_w, list_b = [], []
epochs = range(15)
losses = []
for epoch in epochs:
  list_w.append(w.numpy())
  list_b.append(b.numpy())
  current_loss = train(ys, learning_rate=0.1)
  losses.append(current_loss)
  print(f'Epoch {epoch}: w={list_w[-1]:.2f} b={list_b[-1]:.2f}, loss={current_loss:.5f}')
```

→ مقادیر w و b رو ذخیره کنه

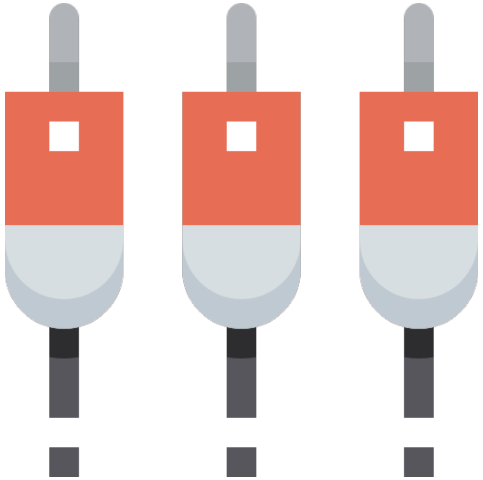
نمایش مقادیر w و b

```
plt.plot(epochs, list_w, 'r',epochs, list_b, 'b')
plt.plot([TRUE_w] * len(epochs), 'r--',[TRUE_b] * len(epochs), 'b--')
plt.legend(['w', 'b', 'True w', 'True b'])
plt.show()
```



نمایش پیشبینی ها

```
test_inputs = tf.random.normal(shape=(NUM_EXAMPLES,))
test_outputs = test_inputs * TRUE_w + TRUE_b


predicted_test_outputs = w*test_inputs + b
plot_data(test_inputs, test_outputs, predicted_test_outputs)
```



تعریف لیست وزن ها

w

```
weights_list = [{'name': "w",
                 'values': list_w
                },
                {
        'name': "b",
        'values': list_b
}]
```

b

[المان ۱ , المان ۲]

دیکشنری   دیکشنری

$\{K_1 : V_1 , K_2 : V_2\}$

"name"   "w"   value list_w

نمایش تغییرات Loss نسبت به w و b

```
def plot_loss_for_weights(weights_list, losses):
  for idx, weights in enumerate(weights_list):
    plt.subplot(120 + idx + 1)
    plt.plot(weights['values'], losses, 'r')
    plt.plot(weights['values'], losses, 'bo')
    plt.xlabel(weights['name'])
    plt.ylabel('Loss')

  plt.show()
```

list_w

w

losses

خلوت گزیده را به تماشا چه حاجت است
چون کوی دوست هست به صحرا چه حاجت است

tf.GradientTape fit

من ✓

خانم مرادی

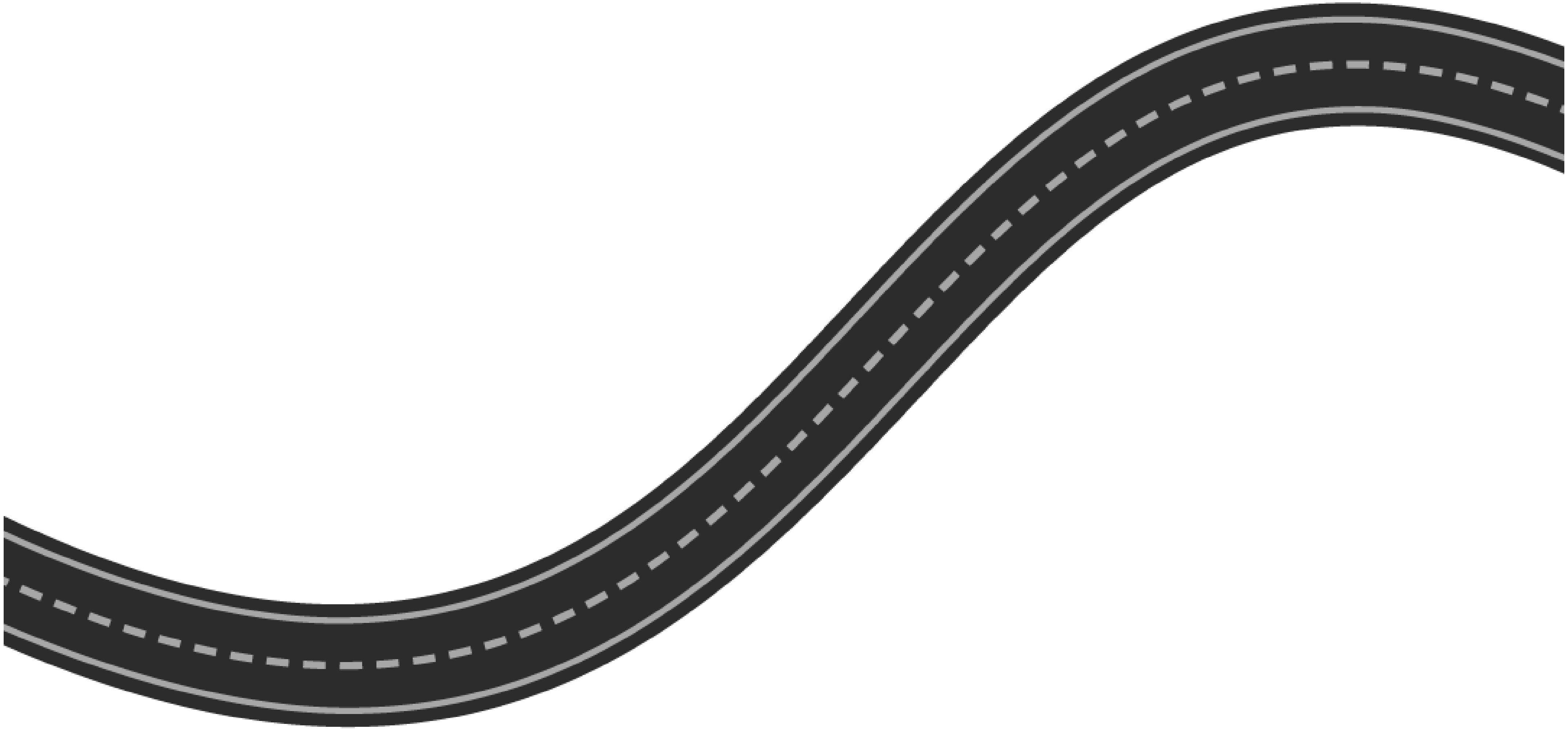
رضا قادری

پرستار

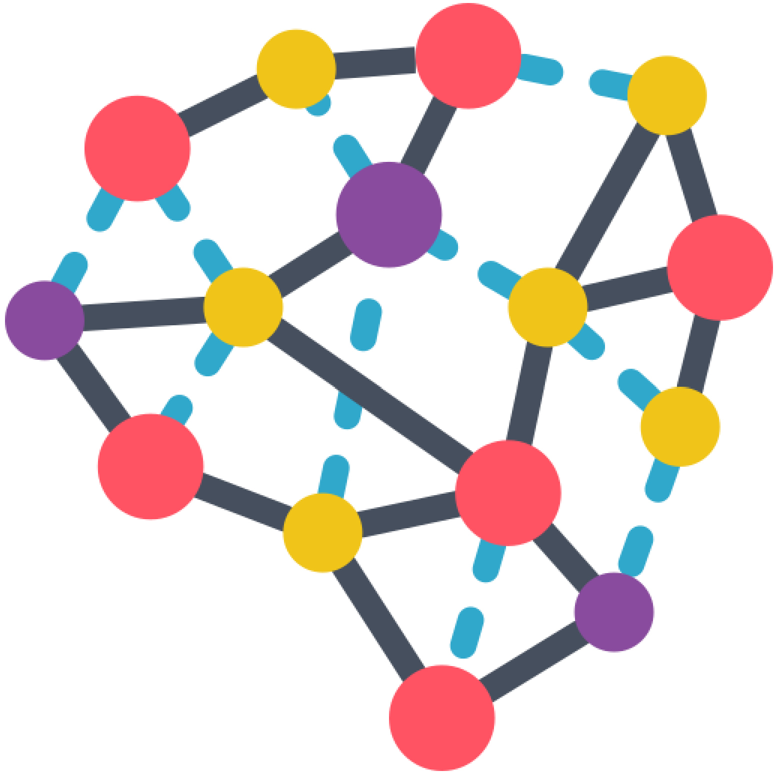
gradiant clipping ?

?

چه گفته ایم و چه خواهیم گفت:

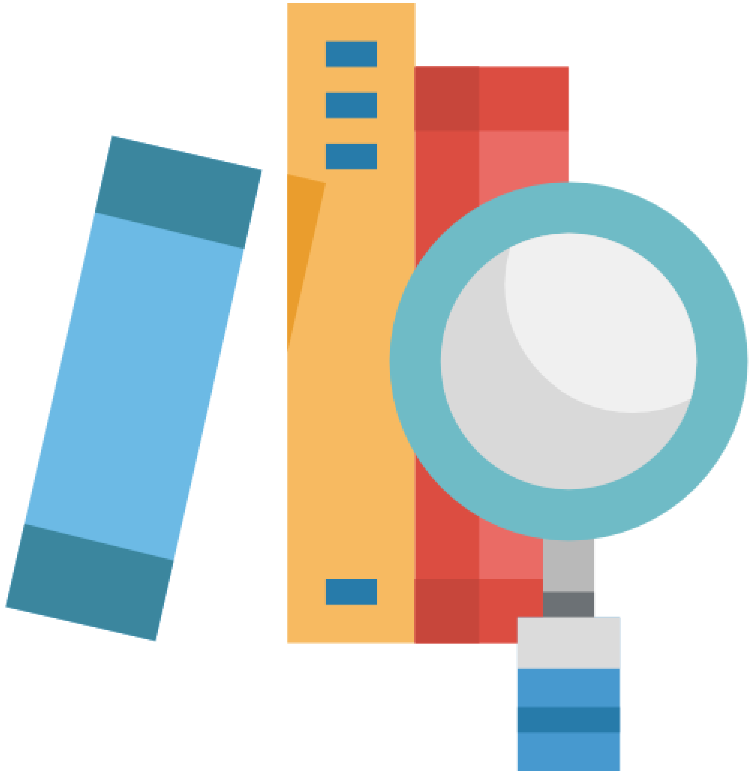


شبکه عصبی با استفاده از tf.GradientTape



اضافه کردن کتابخانه ها

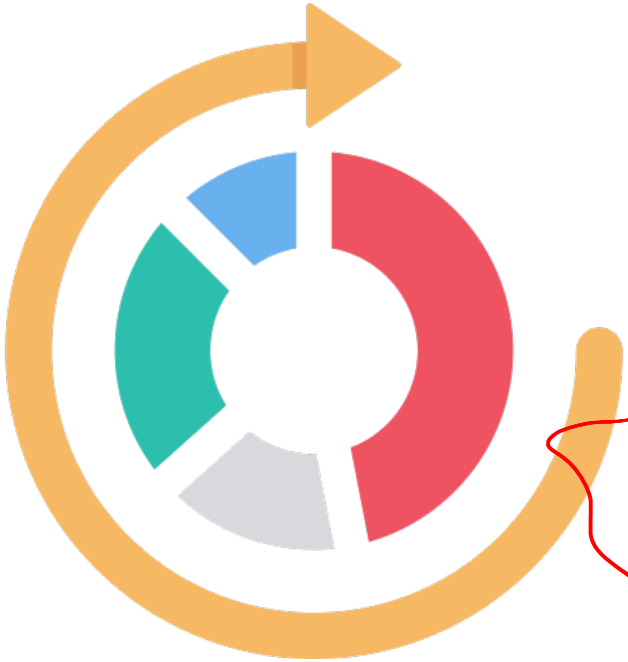
```
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
import tensorflow as tf
import time
```



خواندن داده ها و پیش پردازش

(60000, 28, 28, 1)

(60000, 28, 28) تا



```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = (x_train / 255).reshape((-1, 28, 28, 1))
x_test = (x_test / 255).reshape((-1, 28, 28, 1))

y_test = tf.keras.utils.to_categorical(y_test, 10)
y_train = tf.keras.utils.to_categorical(y_train, 10)
```

one hot encoding

تعریف پارامترها

۲۰ هزار داده ✓

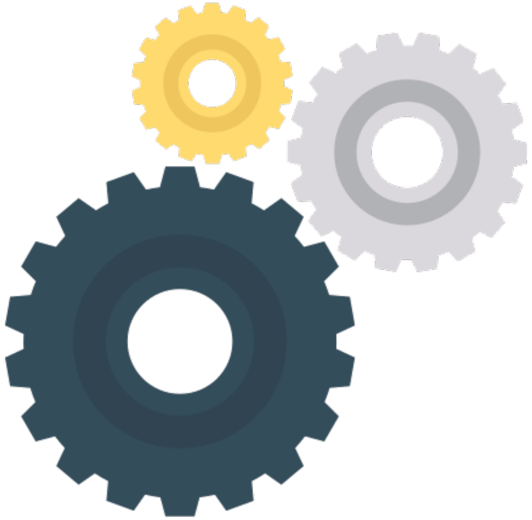
هر batch ✓
128

← چند batch

epochs ۲۰۰۰۰
$$\frac{20000}{128}$$

$$\frac{10006}{128}$$

```
batch_size = 128

epochs = 5

optimizer = Adam(learning_rate=0.01, decay = 0.01/epochs)
```



معماری شبکه

```
model = models.Sequential([
                          layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
                          layers.Conv2D(64, (3, 3), activation='relu'),
                          layers.MaxPooling2D((2, 2)),
                          layers.Dropout(0.25),
                          layers.Flatten(),
                          layers.Dense(128, activation='relu'),
                          layers.Dropout(0.5),
                          layers.Dense(10, activation="softmax")
                          ])
```

تابع step

داده برچسب

```
def step(X, y):
    with tf.GradientTape() as tape:
        y_prime = model(X)
        model_loss = tf.keras.losses.categorical_crossentropy(y, y_prime)

    model_gradients = tape.gradient(model_loss, model.trainable_variables)
    optimizer.apply_gradients(zip(model_gradients, model.trainable_variables))
```

Linear Regression → 1100 → خروجی

همه را → یک بار آپدیت میکنه

20000 → یکی → loss → وزن آپدیت

batch → loss → وزن آپدیت

```
for epoch in rang(30):
    for bn in [تعداد] → 100 → len(X(train)) // batch_size
        batch
```

Train شبکه

```python
bat_per_epoch = int(len(x_train) / batch_size)
for epoch in range(epochs):
    epochStart = time.time()
    for i in range(0, bat_per_epoch):
        start = i* batch_size
        end = start + batch_size
        step(x_train[start:end], y_train[start:end])

    epochEnd = time.time()
    elapsed = (epochEnd - epochStart) / 60.0
    print("epoch: {} took {:.4} minutes".format(epoch, elapsed))
```

نمایش دقت خروجی شبکه

Test

```
model.compile(optimizer=optimizer, loss="categorical_crossentropy", metrics=['accuracy'])

print('Accuracy:', model.evaluate(x_test, y_test, verbose=0)[1])
```

اهمی نمان : نمودار Loss ؟

برای راحتی تابع step را برای یک epoch می نویسیم.

```
def one_epoch_train():
    for i in range(0, bat_per_epoch):
        start = i * batch_size
        end = start + batch_size
        with tf.GradientTape() as tape:
            y_prime = model(x_train[start:end])
            model_loss = tf.keras.losses.categorical_crossentropy(y_train[start:end], y_prime)

        model_gradients = tape.gradient(model_loss, model.trainable_variables)
        optimizer.apply_gradients(zip(model_gradients, model.trainable_variables))
```

تغییرات لازم را اعمال میکنیم :

```
def one_epoch_train():
    step_loss = []
    ...
        with tf.GradientTape() as tape:
            y_prime = model(x_train[start:end])
            model_loss = ...

        step_loss.append(model_loss)
   ...
    return np.mean(step_loss)
```

امین بمانی : نمودار Loss داده های Test هم میخواهیم.

```
def perform_validation():
    val_loss = []
    for i in range(0, bat_per_epoch_test):
        start = i* batch_size
        end = start + batch_size
        y_pred = model(x_test[start:end])
        loss_on_test_data = categorical_crossentropy(y_test[start:end], y_pred)
        val_loss.append(loss_on_test_data)

    return np.mean(val_loss)
```

Metric ها چطور ؟

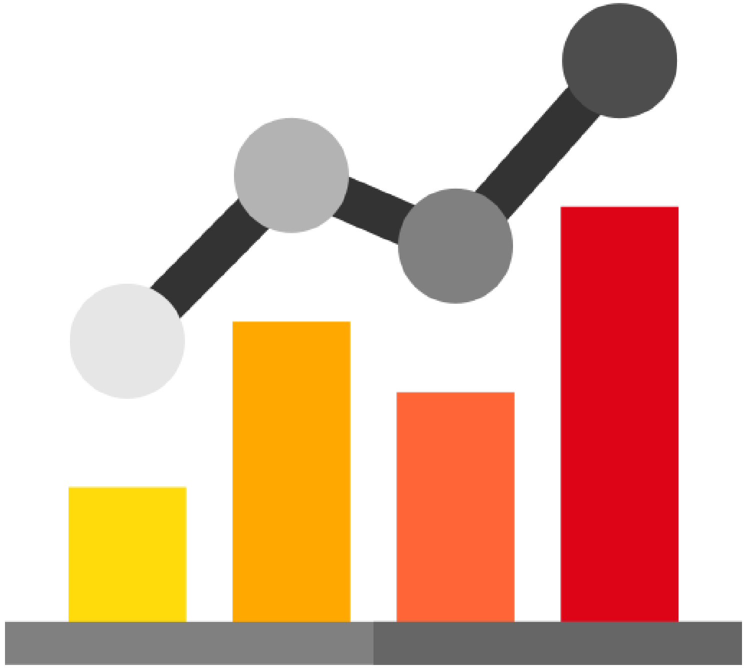
ابتدا یک مثال ساده:

```
from tensorflow.keras import metrics


m = metrics.CategoricalAccuracy()
y_true = [[1, 0, 0], [0, 0, 1]]
y_pred = [[0.1, 0.8, 0.1], [0.1, 0.1, 0.8]]
m.update_state(y_true, y_pred)


print("acc:", m.result().numpy())
```

متد reset

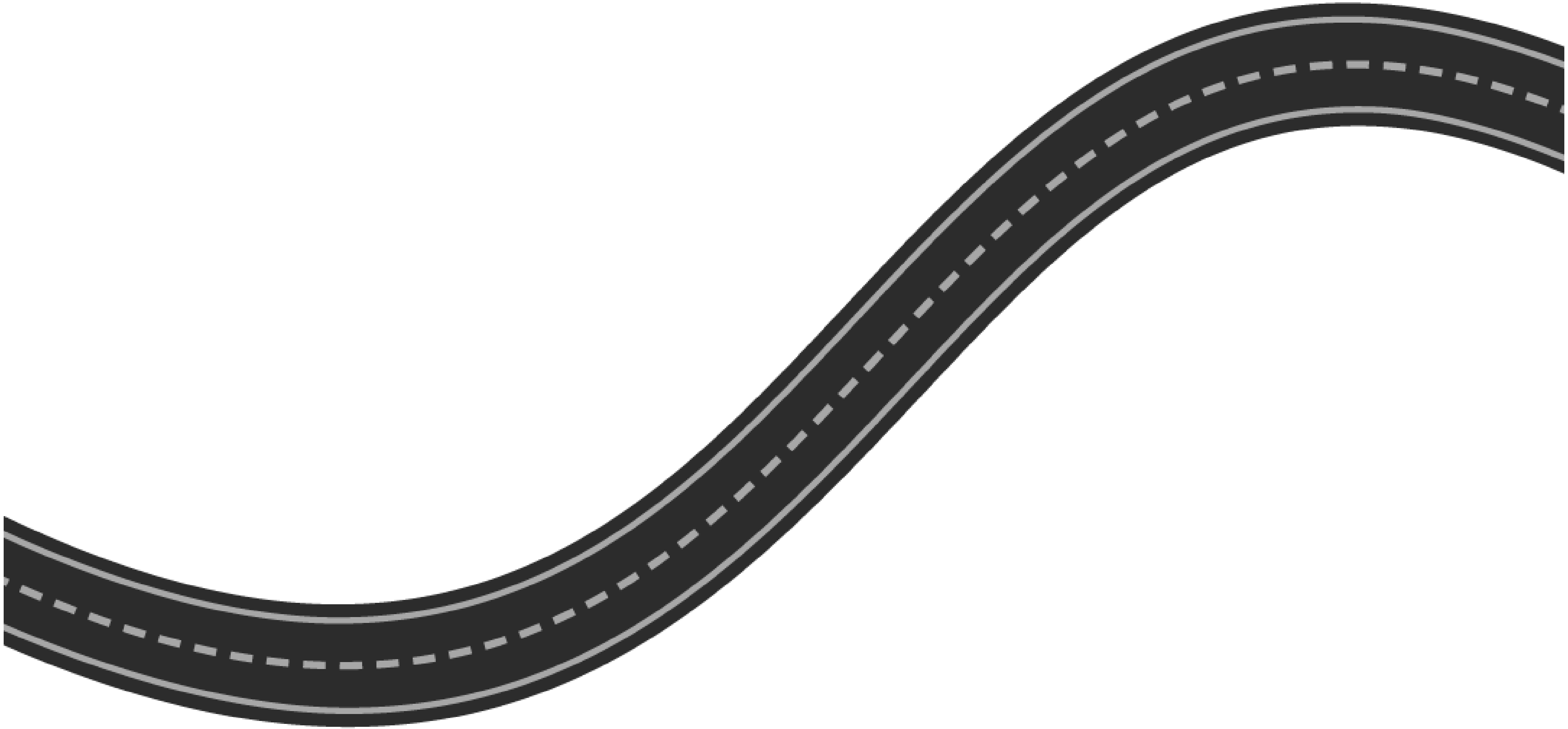
```
m.reset_state()
print("acc:", m.result().numpy())
```

همین ها رو در کد پیاده سازی کنیم !



```
train_metric = metrics.CategoricalAccuracy()
test_metric = metrics.CategoricalAccuracy()

train_metric.update_state(y_train[start:end], y_pred)

test_metric.update_state(y_test[start:end], y_pred)

train_acc = train_metric.result().numpy()
test_acc = test_metric.result().numpy()
```

چه گفته ایم و چه خواهیم گفت:

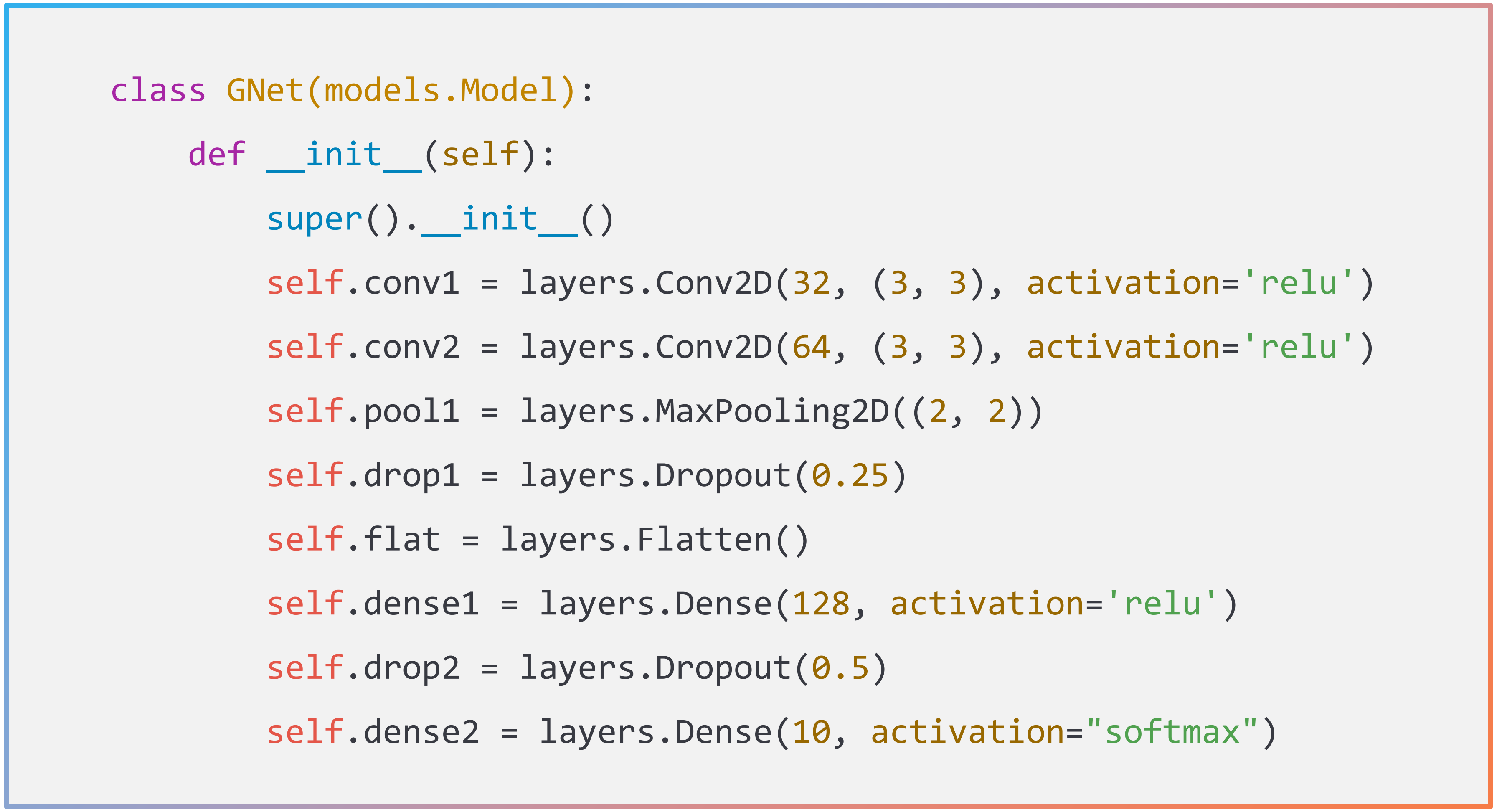


در نظربازی ما بی‌خبران حیرانند
من چنینم که نمودم دگر ایشان دانند

عاقلان نقطه پرگار وجودند ولی
عشق داند که در این دایره سرگردانند

Model Subclassing

متد init

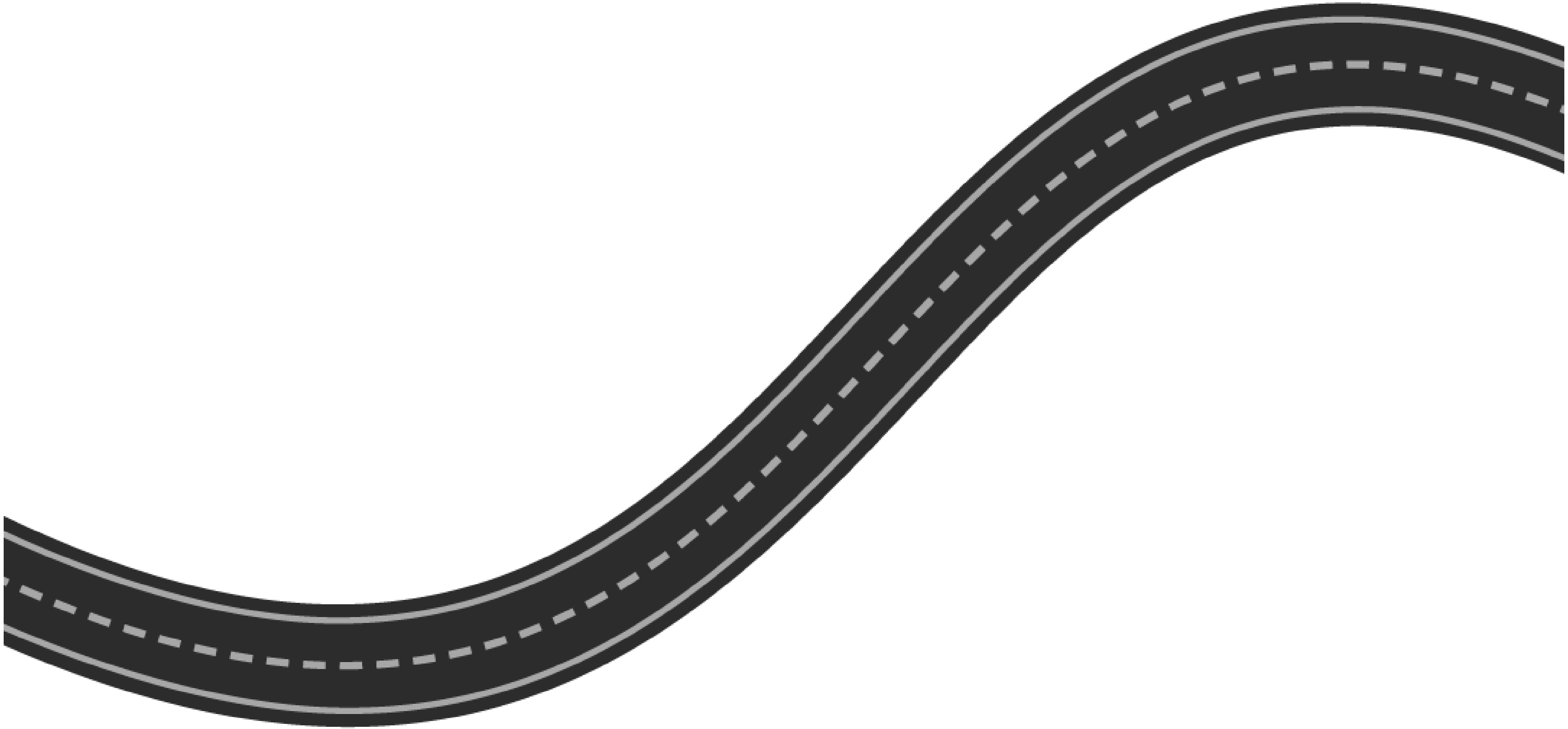


```
class GNet(models.Model):
    def __init__(self):
        super().__init__()
        self.conv1 = layers.Conv2D(32, (3, 3), activation='relu')
        self.conv2 = layers.Conv2D(64, (3, 3), activation='relu')
        self.pool1 = layers.MaxPooling2D((2, 2))
        self.drop1 = layers.Dropout(0.25)
        self.flat = layers.Flatten()
        self.dense1 = layers.Dense(128, activation='relu')
        self.drop2 = layers.Dropout(0.5)
        self.dense2 = layers.Dense(10, activation="softmax")
```

متد call

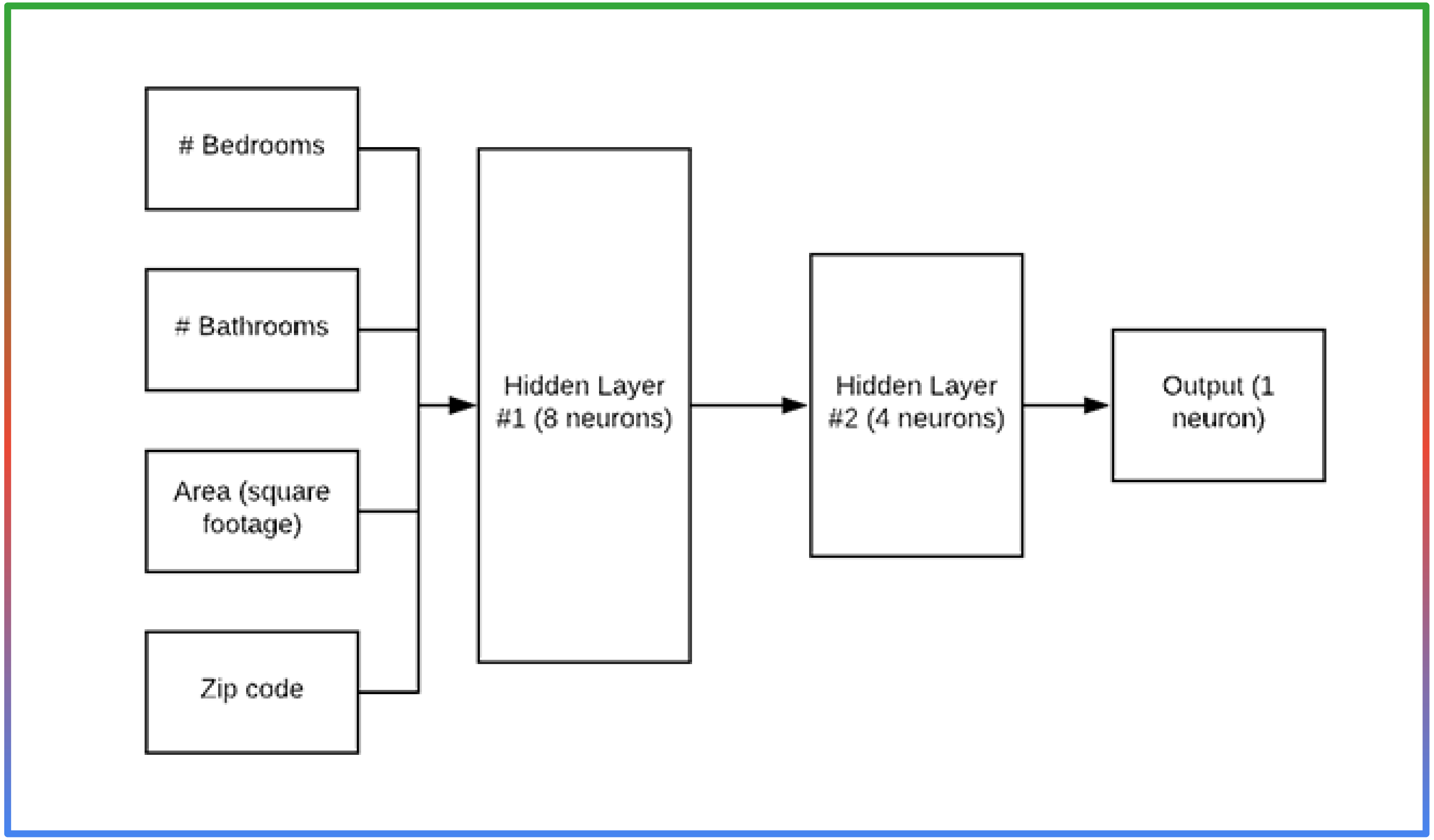
```
def call(self, inputs):
    x = self.conv1(inputs)
    x = self.conv2(x)
    x = self.pool1(x)
    x = self.drop1(x)
    x = self.flat(x)
    x = self.dense1(x)
    x = self.drop2(x)
    x = self.dense2(x)

    return x
```

چه گفته ایم و چه خواهیم گفت:

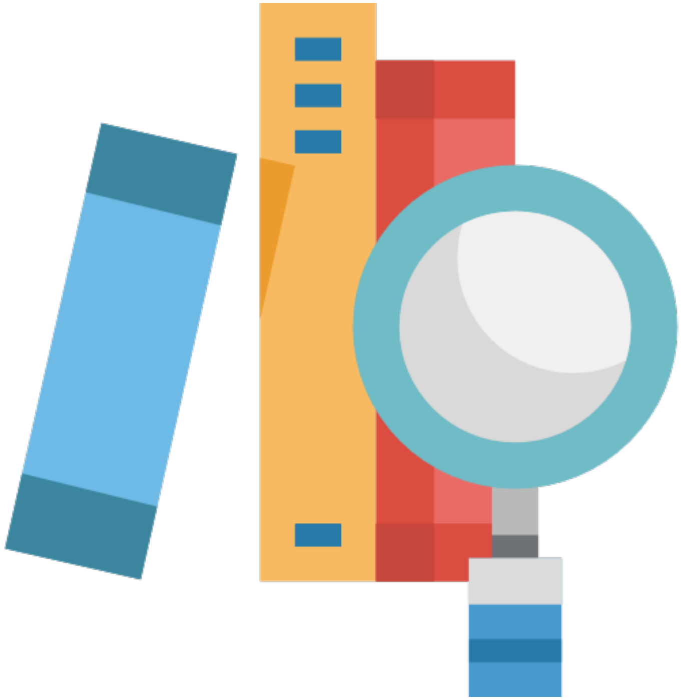


حل یک مثال کامل

ساختار شبکه عصبی



اضافه کردن کتابخانه های مورد نیاز



```
import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import LabelBinarizer

from sklearn.model_selection import train_test_split

from tensorflow.keras import layers, models

from tensorflow.keras.optimizers import Adam
```

استخراج کدپستی های یکتا

```
def load_house_attributes(inputPath):
    cols = ["bedrooms", "bathrooms", "area", "zipcode", "price"]
    df = pd.read_csv(inputPath, sep=" ", header=None, names=cols)
    zipcodes = df["zipcode"].value_counts().keys().tolist()
    counts = df["zipcode"].value_counts().tolist()
```

حذف خانه ها با کدپستی های کم

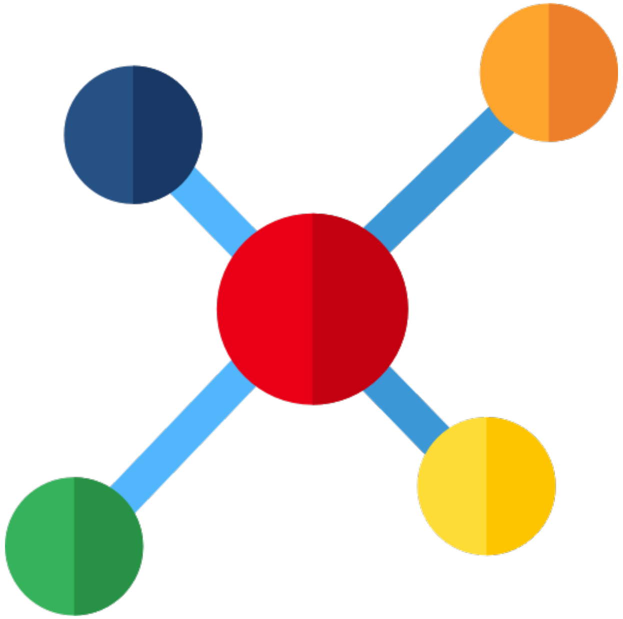
```
for (zipcode, count) in zip(zipcodes, counts):
    if count < 25:
        idxs = df[df["zipcode"] == zipcode].index
        df.drop(idxs, inplace=True)
return df
```

فراخوانی تابع و ایجاد داده های train و test

```
df = load_house_atrributes("HousesInfo.txt")

train, test = train_test_split(df, test_size=0.25, random_state=42)
```

نرمالایز کردن ویژگی های پیوسته

```python
def preprocess_house_attribute(df, train, test):

    continuous = ["bedrooms", "bathrooms", "area"]

    sc = MinMaxScalar()

    trainContinuous = sc.fit_transform(train[continuous])

    testContinuous = sc.fit_transform(test[continuous])
```



One Hot کردن کدپستی

```
zipBinarizer = LabelBinarizer().fit(df["zipcode"])

trainCategorical = zipBinarizer.transform(train["zipcode"])

testCategorical = zipBinarizer.transform(test["zipcode"])

trainX = np.hstack([trainCategorical, trainContinuous])

testX = np.hstack([testCategorical, testContinuous])

return (trainX, testX)
```

تقسیم بندی داده ها و نرمالایز کردن قیمت ها

```
maxPrice = train["price"].max()

trainY = train["price"] / maxPrice

testY = test["price"] / maxPrice
```

ایجاد شبکه عصبی

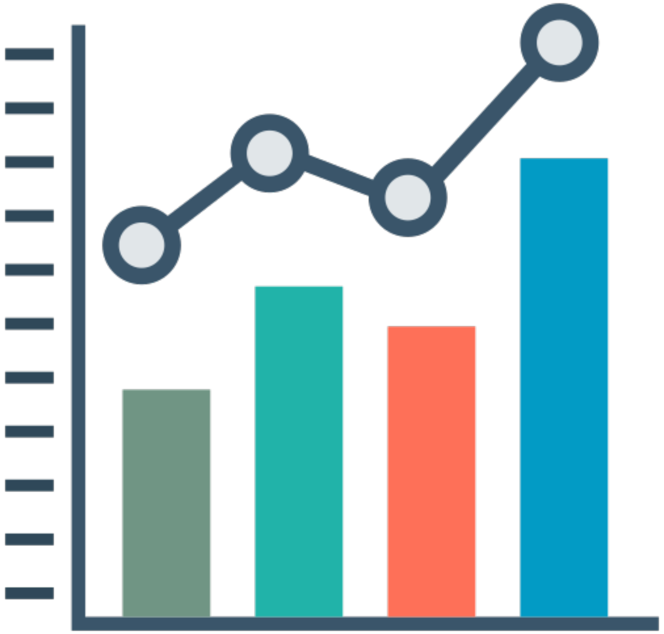
```python
def create_mlp(dim):
    model = models.Sequential([
                               layers.Dense(8, input_dim = dim, activation="relu"),
                               layers.Dense(4, activation="relu"),
                               layers.Dense(1, activation="linear")
                               ])
    return model
```

تنظیم ویژگی ها و Train شبکه

```
model = create_mlp(trainX.shape[1])

opt = Adam(lr=1e-3)

model.compile(loss="mean_absolute_percentage_error", optimizer=opt)

model.fit(x=trainX, y=trainY,validation_data=(testX, testY),

          epochs=200, batch_size=8)
```

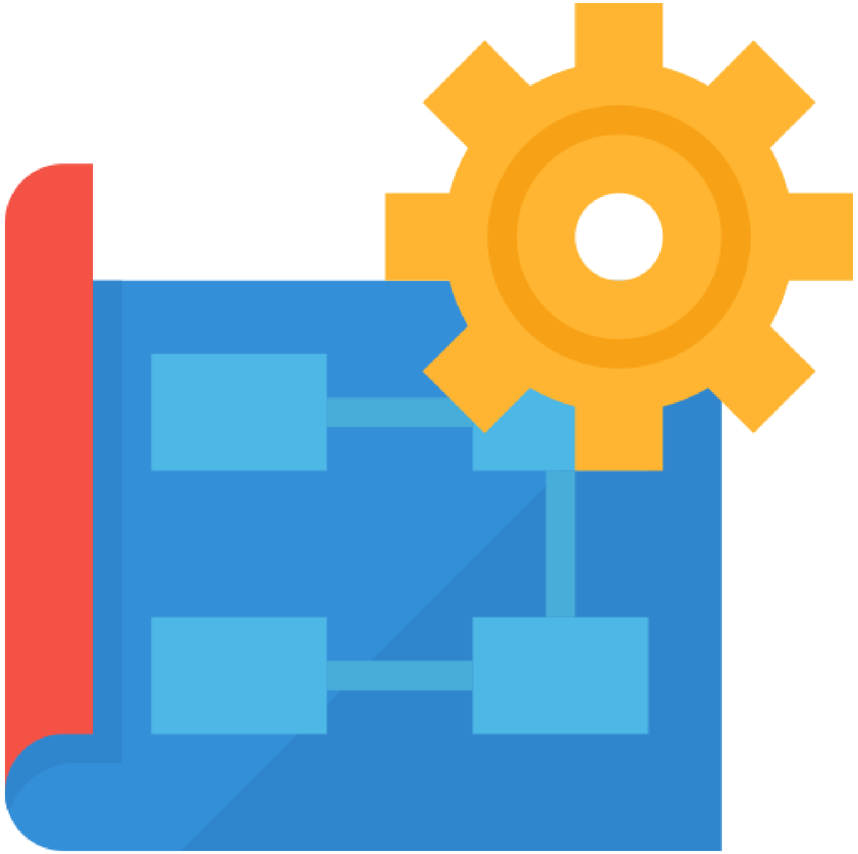
$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

نمایش نتایج در خروجی



```
preds = model.predict(testX)

diff = preds.flatten() - testY

percentDiff = (diff / testY) * 100

absPercentDiff = np.abs(percentDiff)

mean = np.mean(absPercentDiff)

std = np.std(absPercentDiff)

print("[INFO] mean: {:.2f}%, std: {:.2f}%".format(mean, std))
```

تمرین این هفته: همه چیز را از صفر پیاده سازی کنید.



چه گفته ایم و چه خواهیم گفت: